1-7-2020

Prediction of On-Disk Velocity Across a Coaxial Rotor with XGBoost

Ethan Genter
Washington University in St. Louis

Cory Seidel
Washington University in St. Louis

David Peters
Washington University in St. Louis

Washington University in St. Louis

James McKelvey School of Engineering

Washington University Center for Computational Fluid Mechanics

Prediction of On-Disk Velocity Across a Coaxial Rotor with XGBoost

Ethan Genter
e.genter@wustl.edu
Undergraduate Student
Washington Universtiy in St. Louis
St. Louis, MO, USA

Cory Seidel
cory.seidel@wustl.edu
PhD Student
Washington Universtiy in St. Louis
St. Louis, MO, USA

Dr. David Peters
dap@wustl.edu
McDonell Douglas Professor
Washington Universtiy in St. Louis
St. Louis, MO, USA

Contents

List of Figures

List of Tables

1 Abstract

Recent updates in finite state inflow models to solve multi-rotor systems has come at the expense of extra computation time requirements, especially for higher harmonic cases. A potential solution to counter the lengthy time requirements is the application of machine learning algorithms to fit to velocity distributions and predict future distributions. In this paper, we look at XGBoost™as a potential application of machine learning to predict accurate velocity distributions across the rotor disk.

2 Introduction

In this paper, the application of gradient-boosted trees with data from finite-state models for a coaxial rotor system is explored to demonstrate their performance and potential as a future complementary method with finite-models. Currently, finite-state models are very efficient for single-lifting rotors and coaxial rotor systems. However, in higher harmonic solutions (N > 9), the simulation time for a coaxial rotor system increases significantly, in part due to the reverse time marching requirements of the adjoint variables. The application of gradient boosted trees with XGBoost™allows for generation of more densely populated data sets and a greater number of systems by developing data for systems with new parameter combinations. This work will look at the velocity components on each rotor disk with respect to the non-dimensional radial location on the disk ($\bar{r}$) and the following system parameters: rotor spacing (d), rotor solidity (s), Lock Number (γ), and the flapping frequency (p).

Within XGBoost™, there are several parameters that adjust how the trees are formed and perform. This work focuses on the parameters that impact the number of trees (n_estimators), depth of trees (max_depth), how fast they learn (learning_rate), and regularization terms (reg_lambda). Several thousand combinations of these parameters are tested to determine the best set of parameters for fitting the data and make sure that the model is not overfit. It is necessary initially to utilize a larger number of parameters in order to determine the applicable ranges for the parameters. This work focuses on regression analysis and leverages the variability in applicable loss functions and regularization terms within XGBoost™to obtain a better fit model.

Within the realm of machine learning for helicopter applications, a large portion of the research focus has been on the control—especially through reinforcement learning [1][2][3] and autonomous flight—with aerobatic maneuvers being one of the key focuses [4][5]. However, other overlaps with machine learning for helicopter applications exist as well, including the work by Pawar and Jung [6] which focuses on the use of support vector machines to predict damage to composite helicopter rotor blades and how much fatigue life is left in the blades. Kumar and Ben-Tzvi [7] use machine learning for estimating ambient air turbulence for RC helicopters.

Outside of the helicopter realm, machine learning is being utilized in almost every field imaginable. In data sciences one particular framework, XGBoost™, is gaining a lot of attention for its ability to perform efficiently and accurately. XGBoost™is a gradient boosted tree framework that is scalable for large and small data sets [8]. XGBoost™is used throughout a wide range of applications including short-term load forecasting of power systems [9][10], price forecasting [11], and image classification [12].

3 Methodology

XGBoost and Gradient Boosted Trees

XGBoost uses an extreme gradient boosted trees approach with a regression based approach to model a data set. Gradient boosted trees sequentially trains trees with increasing weights on trends that are not fit well and decreasing the weights on those that are already modeled well. This method combines a group of weak learners (short trees) into a strong learner as opposed to building a single large tree. The overall model is controlled by parameters describing how the tree is built and how many trees are desired. In this work we focus on using n_estimators, max_depth, learning_rate, and reg_lambda.

In XGBoost, n_estimators controls how many trees will be used to fit the data. While the number of trees needed is dependent on the other parameters within the model, n_estimators eventually reaches a point where adding more does not increase the performance and too many trees can lead to overfitting. Overfitting occurs when the model fits the training data too closely and performance on the testing data decreases. A basic example of overfitting is if the model fits a point that is a clear outlier to the rest of the data set.

The parameter max_depth dictates how deep each individual tree is allowed to be. This generally should be a lower number because the purpose of gradient boosted trees is to use a combination of weak learners, which are shorter trees. As will be shown in the results section, there is a trade-off in n_estimators and max_depth to reach similar performance. The size of the data set will also factor into what the max_depth parameter should be. A smaller data set generally requires fewer decisions and therefore should have a smaller max_depth. The learning_rate parameter impacts how quickly the model fits to the data where finding the balance between accuracy and efficiency is key. Learning_rate also closely relates to the other parameters, especially n_estimators, in how long it takes for the model to obtain a sufficient fit. This is a less predictable parameter to use and varies significantly with each data set. Reg_lambda is an L2 regularization parameter that was varied to control the complexity of the model and prevent overfitting, if any.

In order to find the best combination of parameters, several hundred combinations of these parameters were simulated to determine which combinations were performing the best. Then more simulations were run with parameters in the realm of the higher performing parameters to make sure the best fits were obtained.

Coaxial Helicopter Dataset

The coaxial helicopter data utilized in this work was developed using finite-state models from Ref. 13 [13]. The end goal is to apply this work to a more sparsely populated data set with higher number of harmonics (N) because the time/computing power required for these is much greater. However, for this work we simply applied the XGBoost model to a lower harmonic case (N=3) as a proof of concept. In this case we are looking at variations in a limited number of parameters that include non-dimensional radial location on the disk ($\bar{r}$) and the following system parameters: rotor spacing (d), rotor solidity (s), Lock Number (γ), and the flapping frequency (p). The primary focus here is to determine model accuracy for variations in $\bar{r}$ and d. Overall the data set contained over 100,000 data points but was trimmed down for some of the models to illustrate performance for future sparsely populated data sets.

4 Results

Prior to examining the effectiveness of different XGBoost parameters in combination to one another, it was important to pick those variables which would be examined. Of primary interest were the effects that learning rate, max depth, and number of trees had on the accuracy of the predicting model, all of which are integral to the robust structure of an a a model built using XGBoost. These variables and their relative impact within the model were examined to determine the parameters of optimal fit. They were then used to compare the performance of the predicted velocities with the actual velocities on the lower rotor for data points that are randomly withheld from the training process, as well as examine the effects of train-test splits and random state variables. Lastly, the determination of performance of prediction for intermediate parameters was analyzed for the radial location on the disk ($\bar{r}$) to determine velocity at different locations on the disk, and to observe the possible effect that location had on performance. All post processing of the simulation data was done in MATLAB.

To analyze the effect that max depth, learning rate, and number of trees had on the predicted velocity values, 300 different combinations of the parameters were run. A 4D plot (seen in Fig. 1) was constructed to compare the L2 normalized error within the predicted and actual velocities of each combination relative to one another. This plot shows that most combinations have L2 error lying well under 100 percent; most combinations were on the order of 1-5 percent error. However, cases of the lowest learning rate 0.001 were particularly bad performers with error ranging from 300 percent to a maximum of approximately 678 percent.

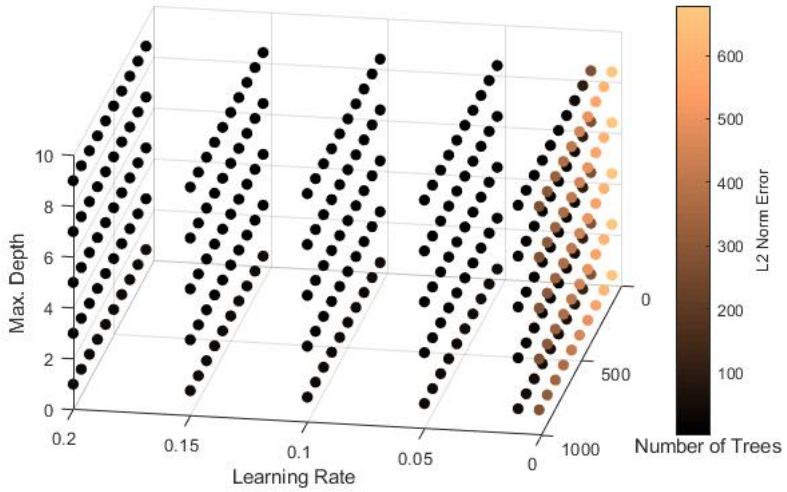

Figure 1: A 4D plot used in comparing combinations of max depth, learning rate, and number of trees to analyze trends of L2 Norm error in predicted versus actual velocity.

To help identify the optimal performers present within the range of combinations the 4D plot seen in Fig. 1 was refined for a smaller range of L2 error. The refined plot is seen in Fig.2 where the color map range has been reduced to 0-1 percent. The process of reducing the color map's range of error (i.e. reducing the displayed range to 0-100 percent, then to 0-50 percent, then to 0-25 percent etc.) showed that generally, combinations of low maximum depth, fewer trees, and higher learning rates (expect for learning rate = 0.001) were less successful. The lowest error identified in Fig. 2 was 0.701 percent, which

indicated optimal parameters. Those parameters were a learning rate of 0.01, maximum depth of 9, and 1000 trees.

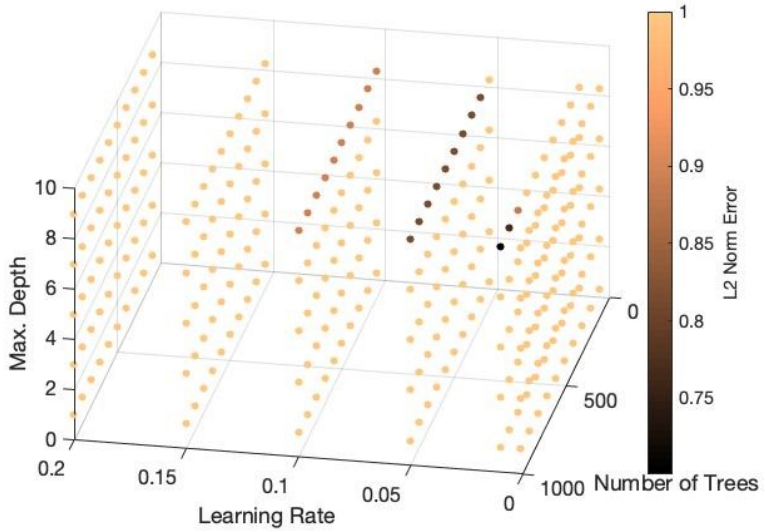

Figure 2: A refined 4D plot used to identify the best combination(s) of max depth, learning rate, and number of trees in reducing L2 normalized error in predicted versus actual velocity.

In addition to examining the effect that different combinations of maximum depth, learning rate, and number of trees had on the L2 error of predicted and actual velocities, other parameters were also explored. The effect of lambda, an XGboost regularization parameter, and eta, an over-fitting parameter were also examined. Both, however, were found to have little effect on the overall model and subsequent L2 error that was observed.

In order to test how sparse the trained data could be, data for intermediate $\bar{r}$ values were tested to compare to the performance of the prediction with the actual values. The model was trained and tested on data where $\bar{r}$ spacing was at increments of 0.05. Combinations of maximum depth, learning rate, and number of trees were test for this model. Due to previous success with a maximum depth of 9 for $\bar{r}$ values at increments of 0.02, only a maximum depth of 9 was explored in these combinations. A 4D plot analyzing the relative L2 normalized error of these combinations predicted velocities can be see in Fig. 3. The trends of present error follow closely to those in the previous model of $\bar{r}$ spacing of 0.02, however, the minimum recorded error was 17.0 percent and the maximum was 674 percent.

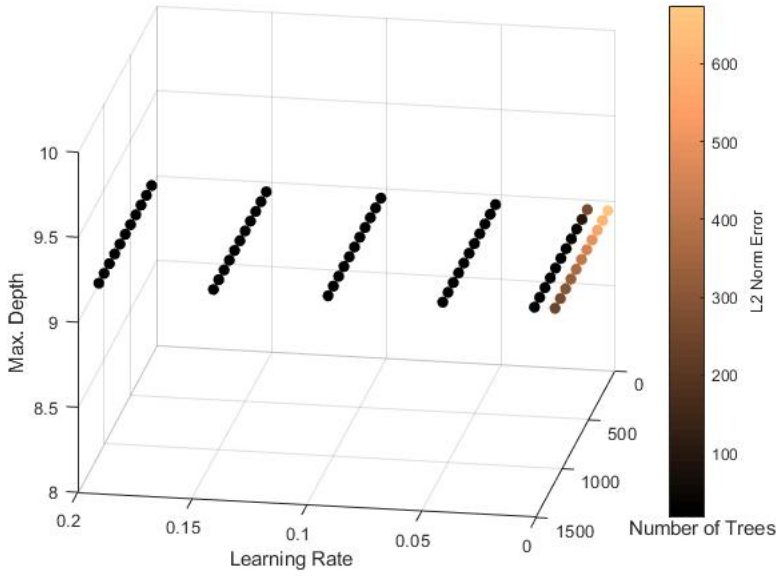

Figure 3: A 4D plot used to analyze the L2 normalized error in testing of intermediate values with an $\bar{r}$ spacing of 0.05.

Beyond the 4D plots the three above parameters were inspected to determine if the trends had converged, also giving a more accurate value of the L2 error. In Fig. 4, for a max depth of 9, different learning rates were evaluated for an increasing number of trees. The model with a learning rate (LR) of 0.001 was developing so slow that its values did not make it into the frame with the other models. It is also notable that model with a LR of 0.01 appears to still be improving and a larger number of trees might be needed for the model to converge. A separate test of greater trees was run with no significant improvement to L2 error in this model, it had already converged at 1000 trees. Error was consistent with the value measured by the 4D graph's minimum.

In Fig. 5, for a learning rate of 0.01, different max depths were evaluated for an increasing number of trees. It is noticeable that the trends for a depth of 1 and 3 are still decreasing. Running tests with more trees would likely lead to the L2 error being closer to the performance of the models that allowed for deeper trees. However, based on the fact that this would likely take a significant number of extra trees, additional tests were not done. Error for a max depth of 9 was consistent with the value measured by the 4D graph's minimum.

In addition to tests run in examining max depth, learning rate, and number of trees more closely. The train-test splits and random state variables of the algorithm were also varied independently. Variation of the random state variable from 111 to 222 and 333 showed no significant change to either Fig. 4 or 5. Adjustment to the train test split showed that increased test size increased L2 error overall and for the best performer. Changing the test size to 50 percent increased L2 error to approximately 0.8 percent and an 80 percent test size increased the error to approximately 1.1 percent.

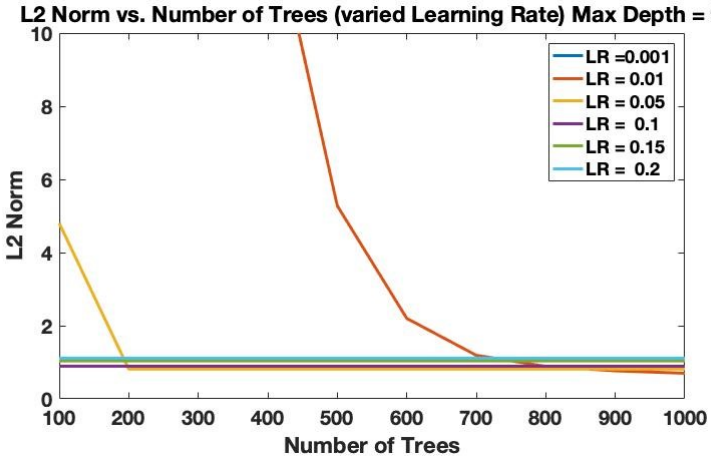

Figure 4: Plotting L2 normalized error at a max depth of 9 versus the number of trees used, for variation in learning rate.

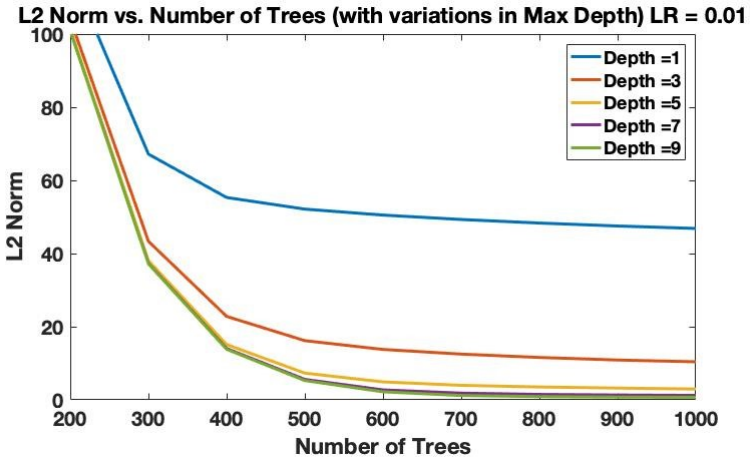

Figure 5: Plotting L2 normalized error at a learning rate of 0.01 versus the number of trees used, for variation in max depth.

Fig. 6 illustrates how the predicted velocity compares to the actual velocity for the model of optimal parameters, as determined by the 4D plot for rotor spacing of 0.02. Values that stray from the red line contain more error, though the overall R-squared value was 0.998.

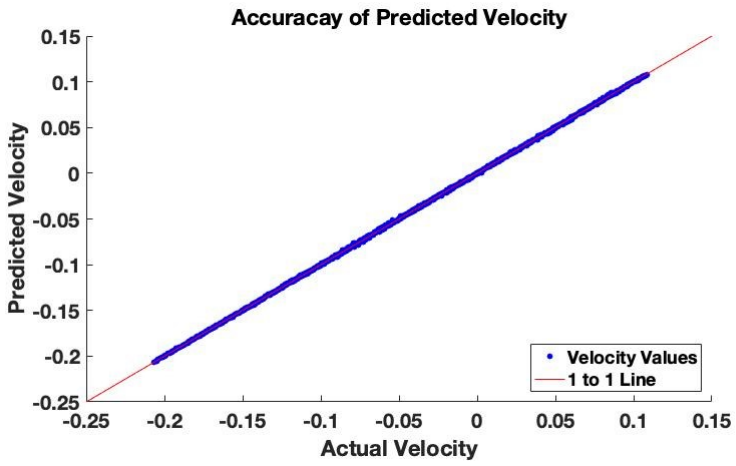

Figure 6: Comparison of predicted and actual velocity values train-test $\bar{r}$ terms for a model with a learning rate = 0.01, max depth = 9, and 1000 trees.

Fig. 7 demonstrates a second approach that was used to look for trends in high error values by comparing the velocity of the actual and predicted velocities to the $\bar{r}$ location. In Fig. 7 this was done for all paramter subsets. Fig. 8 provides a closer look at the performance of the algorithm for one subset of data parameters across the range of $\bar{r}$ values. This shows a really good correlation at all test $\bar{r}$ locations. The actual train and test data, as well as the predicted data all fall along the same curve.

Using the same trained model, data for intermediate $\bar{r}$ values were tested to compare to the performance of the prediction with the actual values. The model was trained on data where $\bar{r}$=0,0.02,0.04...1 and the intermediate values were $\bar{r}$=0.01,0.03,0.05...0.99. The goal being to determine how sparse the trained data points can be while still reaching adequate performance. Fig. 9 shows a large section of velocity terms that are too low. Furthermore, the velocity terms in general are not fitting as tight to the red line as they had in the previous comparison seen in Fig. 6. The large section of velocity terms that are predicted too low are all velocity terms at $\bar{r}$=0.99, which is where the velocity curves fall off due to tip loss.

To analyze this phenomenon more closely, several of the contributing velocity variables were adjusted to more closely observe several different scenarios and locations. Fig. 10 shows for three different cases that in the range of $\bar{r}$=0.9 – 0.99, predicted velocity is consistently less accurate.

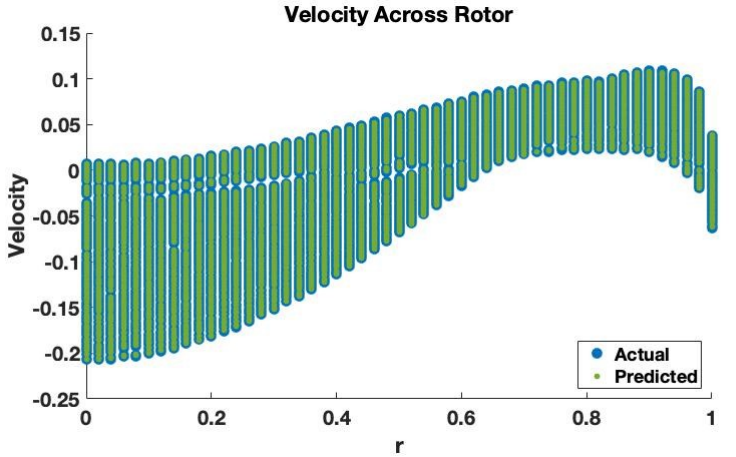

Figure 7: Comparison of actual and predicted velocity values across the rotor for all parameter subsets.

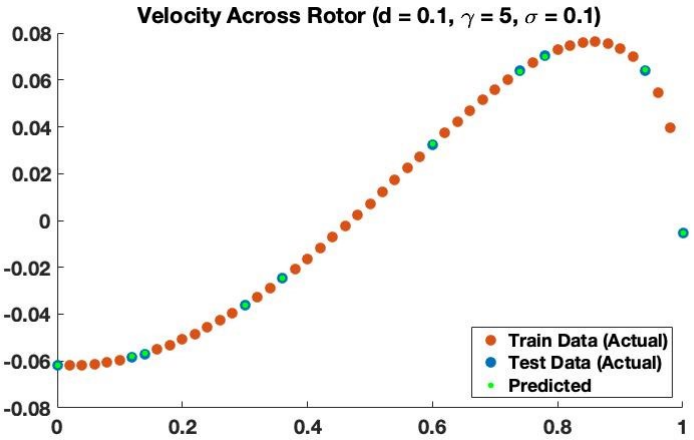

Figure 8: Comparison of actual, train and test velocities, and predicted velocity values across the rotor for a single subset of parameters.

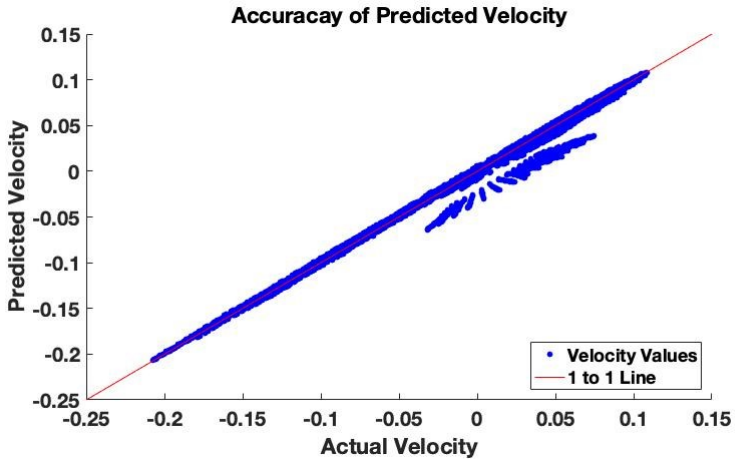

Figure 9: Comparison of predicted and actual velocity of intermediate $\bar{r}$ terms for a model with a learning rate = 0.01, max depth = 9, and 1000 trees.

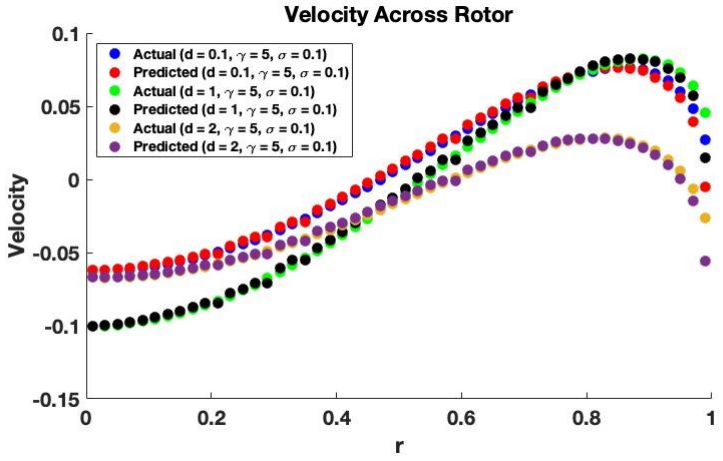

Figure 10: Comparison of actual and predicted velocity values across the rotor for several subsets of parameters.

5 Discussion

Paramount to the study and use of XGBoost within machine learning algorithms is choice of parameters to be implemented, and often varied, within a model. The use of max depth, learning rate, and number of trees are all extremely important within a gradient boosted tree model because they are the fundamental descriptors of a tree and subsequent forest. Initial tests and analysis involved identifying the optimal parameters at which the algorithm would model velocity on the rotor. This of course was monitored through the L2 normalized error resulting from comparison of predicted and actual velocity values across the rotor. The comparison of Fig. 1 and Fig. 2 clearly identifies those proprieties which describe the most successful models. Models of greater depth, lower learning rate, and greater trees had lower L2 error. The exception in this trend is the models of learning rate 0.001 which had particularly high error ranging all the way up to 678 percent. This points to the fact that there is likely a limit to how quickly a model can learn and still develop adequately. Although models of this learning rate performed poorly, those of higher learning rates had error on the order of 1-5 percent. A closer look at Fig. 2 shows that the darkest dot, which would indicate the model of lowest L2 error as per the color bar, lies at a point of max depth equal to 9, 1000 trees, and learning rate of 0.01. The L2 error associated with this point and simulation was 0.701 percent. This point was consistent with the data's trends and quite clearly shows that there is a trade off between learning rate, depth, and the number of trees.

The original rotor spacing used in sampling points to train and test the data was 0.02, and as shown by the minimum recorded error in the optimal model of this spacing, there was good agreement between the predicted and actual velocities. Rotor spacing increments of 0.02 is fairly good, however to test how sparse the data could be and still train and test adequately, a spacing of 0.05 was tested. While the maximum error came down to 674 percent, as seen in Fig. 3 this reduction was erroneous because it still existed within the learning rate of 0.001 and was still very high. More illustrative is the minimum error of the models with a max depth of 9, which was 17.0 percent. This is much higher than the optimal parameters produced in the average 0.02 spacing models, let alone the optimal case. This pointed to training data needing to be quite less sparse than a rotor spacing of 0.05. Subsequently spacings of 0.02 were used.

In addition to a change in rotor spacing, variation in random state variable and train-test split was also explored. Change in train test split provided no significant change in error, while adjustment to train test split did. As excepted, an increase in test size increased the L2 error. Meaning that a reduction in the training split leads to a increase in error. However the increase in error was not as drastic as one might predict. Increasing test size from 20 to 50 percent increased error to just 0.8 percent, and a test size of 80 percent increased error to just 1.1 percent. Of course the size of the data set needs to considered, as an algorithm that has a large data set to work with will still perform well in lower training percentages because the low percent of the large set is still a considerable amount of data. Such was the case with this data set of nearly 100,000 points. A training split of 20 percent would still have nearly 20,000 data points to train off of and would still learn quite well, as was seen.

If we reanalyze the 4D plots used in initial identifying the model's optimal parameters the obvious question is, given the trends, will error converge at just 1000 trees or below, or will it continue to develop. Fig. 4 and Fig. 5 help us answer this question. Fig. 4 shows that a learning rate of 0.001 doesn't register within the figure as it is is developing

too slowly, while learning rates of 0.1, 0.15, and 0.2 all appear as flat lines because they developed very quickly and converged for just 100 trees. Similarly, a learning rate of 0.05 converges for 200 trees. However, the learning rate of 0.01, the optimally identified parameter seems to still be developing beyond 1000 trees. When the number of trees was increased it was observed that the curve had in fact converged as the error was not reduced to a significant degree beyond 1000 trees. Fig. 5 on the other hand shows the development of different max depths with varied number of trees. The higher value depths all converge for 800 trees, while the lower value depths of 1 and 3 are still developing, though they are developing quite slowly. To achieve a similar level of error as the higher value depths the number of trees would likely need to be significantly increased for which computational power would be a premium. For this reason higher number of trees were not explored to achieve convergence at lower depths.

The success of the optimal model found in Fig. 2, and subsequently verified, is illustrated in Fig. 6. All predicted values lie tightly along the one to one line, indicating a strong fit for the model and good accuracy of the predicted velocity values. The R-squared value of 0.998 validates this point.

Fig. 7 demonstrates a second approach that was used to explore the trends in high error valued, predicted velocities. This compared predicted and actual velocities across the rotor (i.e. at respective $\bar{r}$ locations). Fig. 7 shows the comparison of all predicted and actual velocities for all sets of system parameters (i.e. rotor spacing, rotor solidity, Lock Number, and flapping frequency) which contribute to the on disk velocity components. While the presence of all rotor-velocity curves gives a general indication as to the behavior of the velocity across the rotor, Fig. 8 is more descriptive. Fig. 8 compares the actual trained and tested data compared to the predicted velocity values for a singular subset of system parameters. The curve shows again that there is good correlation, across the rotor, between the accuracy of predicted data and the actual train/test data for our optimal model.

However, we observe in Fig. 9 that there are still inaccuracies in the algorithm's predicted velocities. A large section of the data's predicted velocities fall below what is expected by the one to one comparison line when the trained data is tested on intermediate rotor spacing values of 0.02 (i.e. $\bar{r}$=0.01,0.03,0.05...0.99). Although the the training data is still at a rotor spacing of 0.02, and therefor not nearly as sparse as the rotor spacing of 0.05 analyzed previously, it is still experiencing inaccuracies.This large section of inaccurate velocities are found to act at a position $\bar{r}$=0.99. At this location we can see that both Fig. 7 and Fig. 8 show a large drop in velocity; this drop is due to tip loss. To observe this phenomenon for several subsets of system parameters we can see Fig. 10, which as in Fig 8 compares actual and predicted velocities across the rotor. The same inaccuracy is observed at rotor spacings of $\bar{r}$=0.9-0.99 for all subsets, tip loss causes large discrepancy between the predicted and actual velocity. Perhaps indicating that for increased accuracy at the tip the training data needs to be less sparse, which, in turn, would sacrifice speed and computational efficiency.

6 Conclusion

In this paper the application of XGBoost and the use of gradient-boosted trees was used to explore the behavior of data from finite-state models of coaxial rotor systems. To this end the analysis sought not only to demonstrate the models' individual performance

but also investigate the potential for their use in higher harmonic solutions.

In order to carry out this analysis, XGBoost, a gradient-boosted tree algorithm was leveraged to help help predict on-disk velocities of the coaxial rotor system. XGBoost accomplishes this through the use of sequentially trained trees that gain or lose learning weight based on their relative success. Effectively, smaller, less learned trees are combined with one another to create a strong learned forest rather than a singularly strong tree. Several variable parameters are particularly important within XGBoost and were explored within this paper. Primarily, the effect of maximum depth of each tree, learning rate of the algorithm, and number of trees were explored for their effect on the L2 normalized error in actual and predicted velocities. All three of these parameters were varied in 300 different combinations to identify the optimal combination of parameters. Within the regions of the best performing parameters, further simulations were then run to validate the best fit of the data.

Following initial simulation of the 300 combinations it was found that simulations of lower learning rate, greater max depth, and higher number of trees performed the best. Within the window of parameter ranges that was tested, optimal parameters were found to be a max depth of 9, learning rate of 0.01, and 1000 trees. Comparison of the actual velocities with those predicted by the simulation with these parameters led to L2 error of just 0.701 percent, a good fit to the data. This point along with the trends seen in the data demonstrated a clear trade off between learning rate, depth, and number of trees. Further expansion of the number of trees that could be run within a simulation showed that L2 error with respect to the number of trees had converged for the optimal parameters of max depth and learning rate. Those that had not converged would have required a significant number of more trees to achieve convergence and were therefore not explored. The success of this model was re-verified via a direct comparison of actual and predicted velocities along a one-to-one line, this showed a tight fight along the line with an R-squared value of 0.998.

Additional parameters outside of the primary three were explored as well. Eta, a regularization term for over fitting was found to have little to no effect on the simulation. Lambda, a separate regularization termed performed similarly with little to no effect. The random state variable of the model as well as the train-test split were also varied. Change to the random state variable had no significant effect on the L2 error, though change to the train-test split did. As expected a decrease in the train split increased the L2 error of the model, however, even at an 80 percent test split (i.e. a 20 percent train split) the L2 error was only 1.8 percent. This small increase in error was due to the large size of the data set, which had nearly 100,000 data points, meaning the model could still train off of a large amount of data.

A change in the rotor spacing of the sampled data was used to observe how sparsely the data could exist and still train and test adequately. The spacing was changed to 0.05. Under optimal parameters the new model had a significantly higher error, 17 percent, than the that of the one spaced at increments of 0.02. Additional rotor spacing increments were not explored beyond 0.02 and 0.05, though future work should seek to test these spacings as an acceptable error might exist between this range and increase the efficiency of the overall model.

A second approach was used to explore the trends in higher error valued, predicted velocities. The approach compared predicted and actual velocities across the rotor (again the rotor spacing increment used was 0.02). The correlation upon comparison was very good, as can be seen in Fig. 8. However, inaccuracies within the algorithm's predicted

velocities still existed when trained data was tested on intermediate rotor spacings (i.e. $\bar{r}$=0.01,0.03,0.05...0.99 as opposed to $\bar{r}$=0,0.02,0.04...1). A large portion of velocities at a rotor position of $\bar{r}$=0.99 were under-predicted. This under-prediction was due to velocity loss experienced at the tip. This phenomenon was confirmed across multiple subsets of system parameters. This indicated that training data may need to be less sparse in order to account for the rotor positions in the range closer to the tip, $\bar{r}$=0.9-0.99, though this would sacrifice computational efficiency within the model.

In addition to tip loss, future work should explore expansion of the range of max depth and learning rate as trends indicated an increase in max depth and decrease in learning rate (to a certain limit) reduced L2 error. Due to limits on the computational power of this investigation this was not done, however it would aid in continuing to test additional parameter combinations. Furthermore, the sparsity of the data was only briefly explored to define limits at which the model would not perform well, the range within these limits should be analyzed further. Ultimately this analysis served as a proof of concept for the coaxial rotor, lower harmonic case of N=3. Application to higher harmonic solutions should be the next step in these studies. Successful implementation of XGBoost and the use of gradient boosted trees would help to greatly reduce the simulation time necessary at higher harmonics (N > 9).

Bibliography

[1] Rogier Koppejan and Shimon Whiteson. *Neuroevolutionary Reinforcement Learning for Generalized Helicopter Control*. Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation - GECCO '09, July 2009.

[2] J.A. Bagnell and J.G. Schneider. *Autonomous Helicopter Control Using Reinforcement Learning Policy Search Methods*. Proceedings of 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat.No.01CH37164, May 2001.

[3] Dimitris C. Dracopoulos and Dimitrios Effraimidis. *Genetic Programming for Generalised Helicopter Hovering Control*. European Conference on Genetic Programming. Springer, Berlin, Heidelberg, 2012.

[4] Pieter Abbeel, Adam Coates, and Andrew Y. Ng. *Autonomous Helicopter Aerobatics through Apprenticeship Learning*. The International Journal of Robotics Research VOl. 29, (13), pp. 1608-1639, 2010.

[5] P. Abbeel et al. *An Application of Reinforcement Learning to Aerobatic Helicopter Flight*. NIPS'06 Proceedings of the 19th International Conference on Neural Information Processing Systems, December 2006.

[6] Prashant M. Pawar and Sung Nam Jung. *Support Vector Machine Based Online Composite Helicopter Rotor Blade Damage Detection System*. Journal of Intelligent Material Systems and Structures Vol. 19, (10), pp. 1217-1228, 2008.

[7] Anil Kumar and Pinhas Ben-Tzvi. *Extraction of Impact of Wind Turbulence on RC Helicopters Using Machine Learning*. Volume 5A: 40th Mechanisms and Robotics Conference, doi:10.1115/detc2016-59384, 5 December 2016.

[8] Tianqi Chen and Carlos Guestrin. *XGBoost*. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD'16, doi:10.1145/2939672.2939785, August 2016.

[9] Huiting Zheng, Jiabin Yuan, and Long Chen. *Short-term Load Forecasting Using EMD-LSTM Nueral Networks with a XGBoost Algorithim for Feature Importance Evaluation*. Energies Vol 10, (8), pp. 1168, 2017.

[10] Raza Abid Abbasi et al. *Short Term Load Forecasting Using XGBoost*. Workshops of the International Conference on Advanced Information Networking and Applications. Springer, Cham, 2019.

[11] Mesut Gumus and Mustafa S. Kiran. *Crude Oil Price Forecasting using XGBoost*. 2017 International Conference on Computer Science and Engineering (UBMK). IEEE, 2017.

[12] Xudie Ren et al. *A Novel Image Classification Method with CNN-XGBoost Model*. International Workshop on Digital Watermaking. Springer, Cham, 2017.

[13] Cory Seidel. *Coupled Inflow and Structural Dynamics of Rotors with Time Delays and Adjoint Variables*. Washington University in St. Louis, Doctor of Science Thesis, May 2020.